

IRI: What novel interfaces will HPC expose for cross-facility workflows?



Why are we here today?

- HPC has seen a rise in non-traditional use cases:
 - Complex cross-facility workflows
 - Data/Analysis-centric users
 - Machine-to-machine workflows
- Evidence in multiple projects
 - Superfacility/SENSE/Intersect/ACE (OLCF)
 - Blueprints of DOE's Integrated Research Infrastructure (IRI)
- New class(es) of users and use cases
 - Machine/Service/Collab accounts
 - Data/Analysis-centric users
 - HPC integration in data/ML workflows from other facilities
- To service these use cases we need to step out of our “shell” and provide alternative interfaces
 - APIs
 - Workflow orchestrators
 - UIs (Jupyter etc)
- Landscape of interfaces is rather fragmented today
 - DOE's IRI seeks to harmonize this at least for ASCR facilities
 - Opportunity to create interfaces that facilitate cross-facility workflows

Think!

- What would you like alternative interfaces to look like?
- What are the difficulties with your current cross-facility workflow?



Agenda

12:15-12:20 Welcome and opening remarks

Bjoern Enders, National Energy Research Scientific Computing Center (NERSC)



Bjoern Enders
NERSC/LBL



John MacAuley
ESnet

12:20-12:25 Introduction to the IRI interfaces subcommittee and charter

John MacAuley, Energy Sciences Network (ESnet)

12:25-12:50 Lightning Talks

Superfacility API and Jupyter

Bjoern Enders, National Energy Research Scientific Computing Center (NERSC)



Xi Yang
ESnet



Ryan Prout
OLCF

Cross-Facility Science Workflows Automation and Orchestration

Xi Yang, Energy Sciences Network (ESnet)

Services for integrating Data Portals

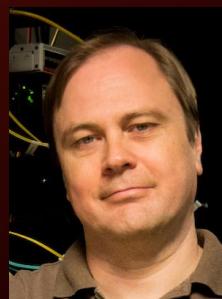
Ryan Prout, Oak Ridge Leadership Computing Facility (OLCF)

Globus Platform Interfaces for Automating Science

Kyle Chard, Argonne National Lab (ANL)

FirecREST: an Open-Source API for HPC

Juan Pablo Dorsch, CSCS



Ilya Baldin
JLAB



Juan Pablo Dorsch
CSCS



Kyle Chard
ANL

12:50 -1:15 Panel discussion and Open Q&A

Please hold your questions until the Q&A



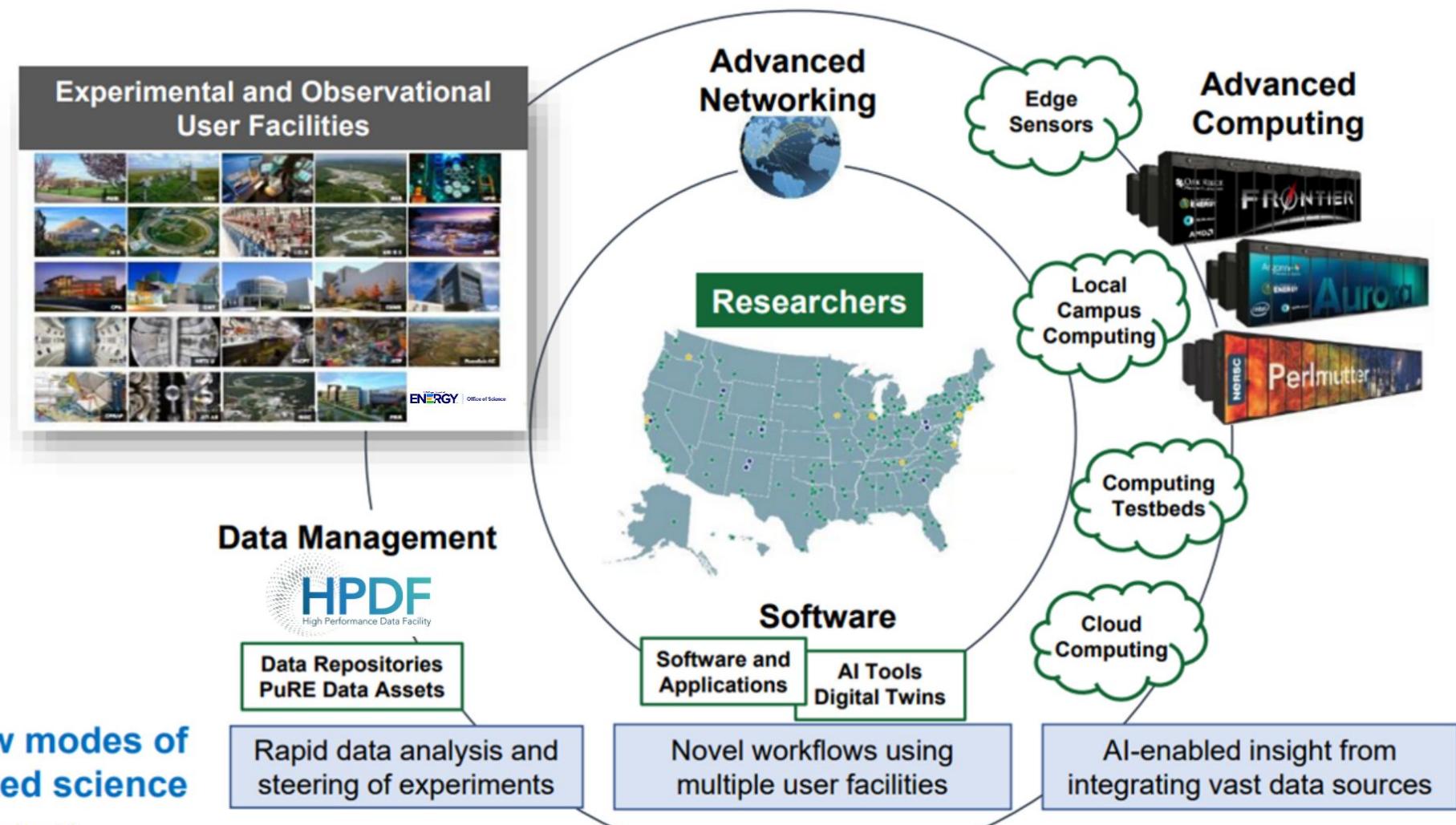
IRI technical subcommittee

John MacAuley
Energy Sciences Network (ESnet)



DOE's Integrated Research Infrastructure (IRI) Vision

To empower researchers to meld DOE's world-class research tools, infrastructure, and user facilities seamlessly and securely in novel ways to radically accelerate discovery and innovation.



Office of Science

<https://iri.science/>



IRI Interfaces Charter

- Gather requirements, define use cases, and specify unified interfaces for the DOE ASCR Facilities.
- Tasked with enhancing the "user experience" for interaction and management of complex workflows across ASCR facilities
 - Goal is to facilitate a more cohesive and efficient research infrastructure.
- Consider applicable standards and build upon the work of other organizations where available to deliver a reference IRI interface implementation.

Launched May 2024

Interfaces Technical Subcommittee

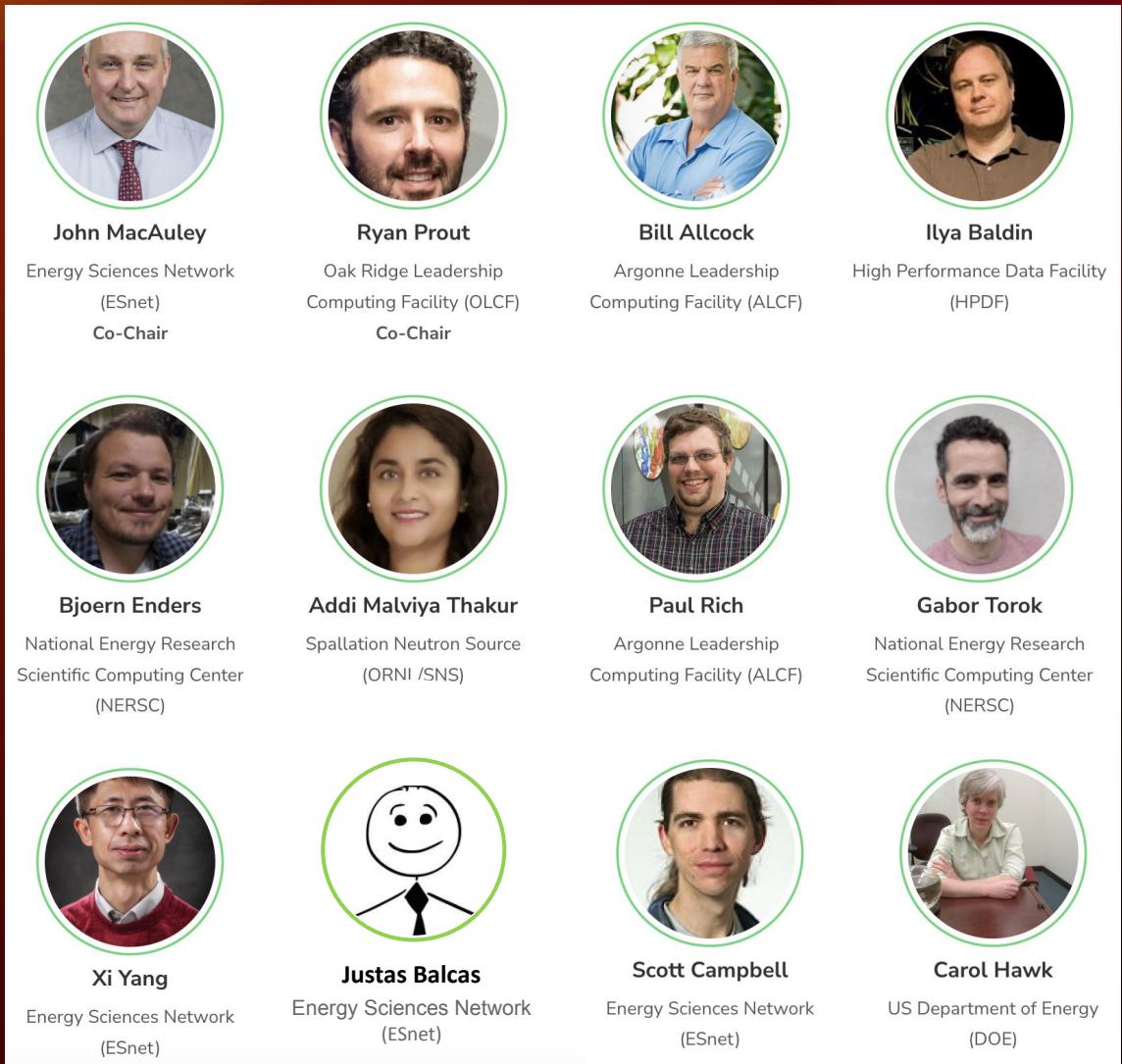
- Design a **minimal functional API** and deploy it at multiple sites
 - Review existing API schemas
 - Propose IRI schema: endpoints, architecture, infrastructure,...
 - Implement MVP
- Explore how to align **Jupyter** across sites

Co-chairs: John MacAuley and Ryan Prout. Members: Bill Allcock, Ilya Baldin, Bjoern Enders, Paul Rich, Addi Thakur, Gabor Torok, Xi Yang



Status

- Team formation complete
 - But always looking for eager participants!
- Delivered
 - Subcommittee charter.
 - 2-year development roadmap.
- Developed a shared understanding
 - Different experiences, technologies, and terminologies.
- Actively defining a Facility Status API
 - Use case and requirements definition.
 - Modelling and protocol definition.
- Cross-pollinating subcommittees
 - TRUSTID, Outreach and Engagement.



Lightning Talk I

Superfacility API and Jupyter

Bjoern Enders
NERSC



NERSC is the Production HPC & Data Facility for DOE Office of Science Research



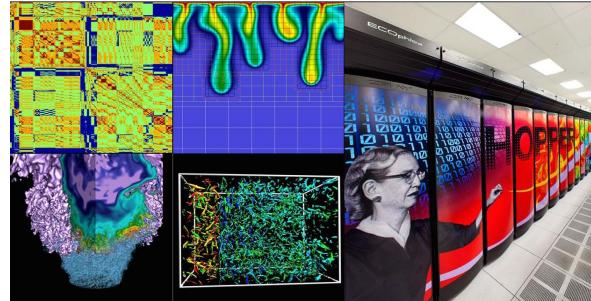
U.S. DEPARTMENT OF
ENERGY

Office of
Science

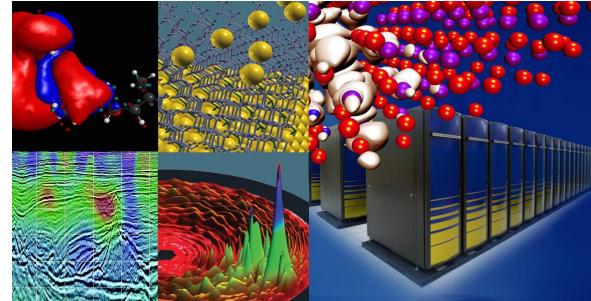
Largest funder of physical
science research in U.S.



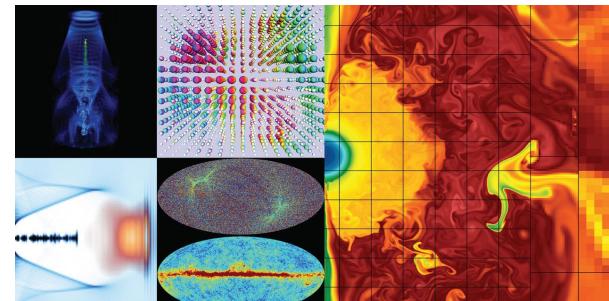
Biology, Energy, Environment



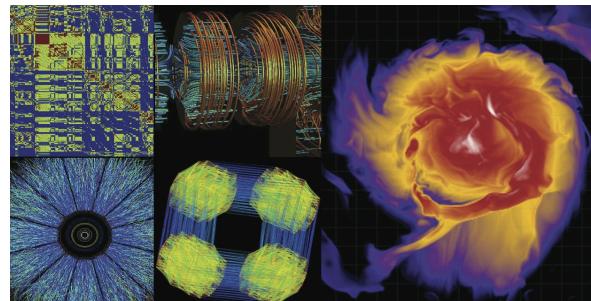
Computing



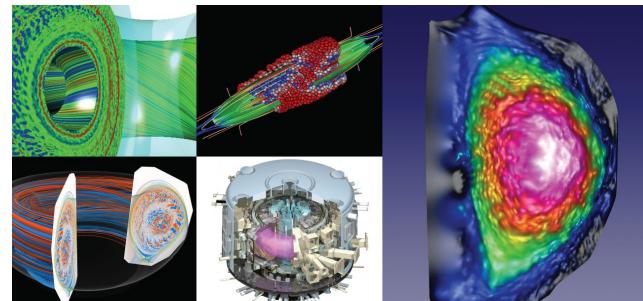
Materials, Chemistry,
Geophysics



Particle Physics,
Astrophysics



Nuclear Physics

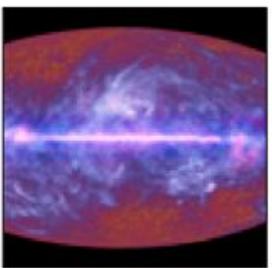


Fusion Energy,
Plasma Physics

NERSC supports a large number of users and projects from DOE SC's experimental and observational facilities



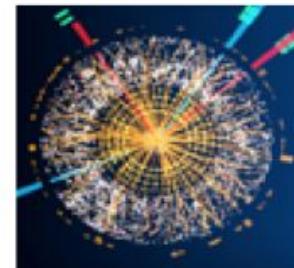
Palomar Transient
Factory
Supernova



Planck Satellite
Cosmic Microwave
Background
Radiation



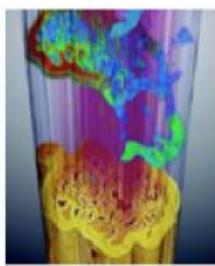
Star
Nuclear Physics



Atlas
Large Hadron Collide



Dayabay
Neutrinos



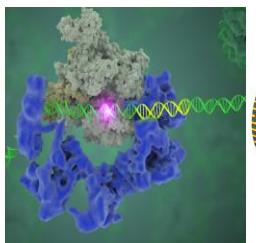
ALS
Light Source



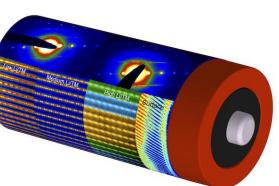
LCLS
Light Source



Joint Genome Institute
Bioinformatics



Cryo-EM



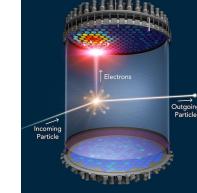
NCEM



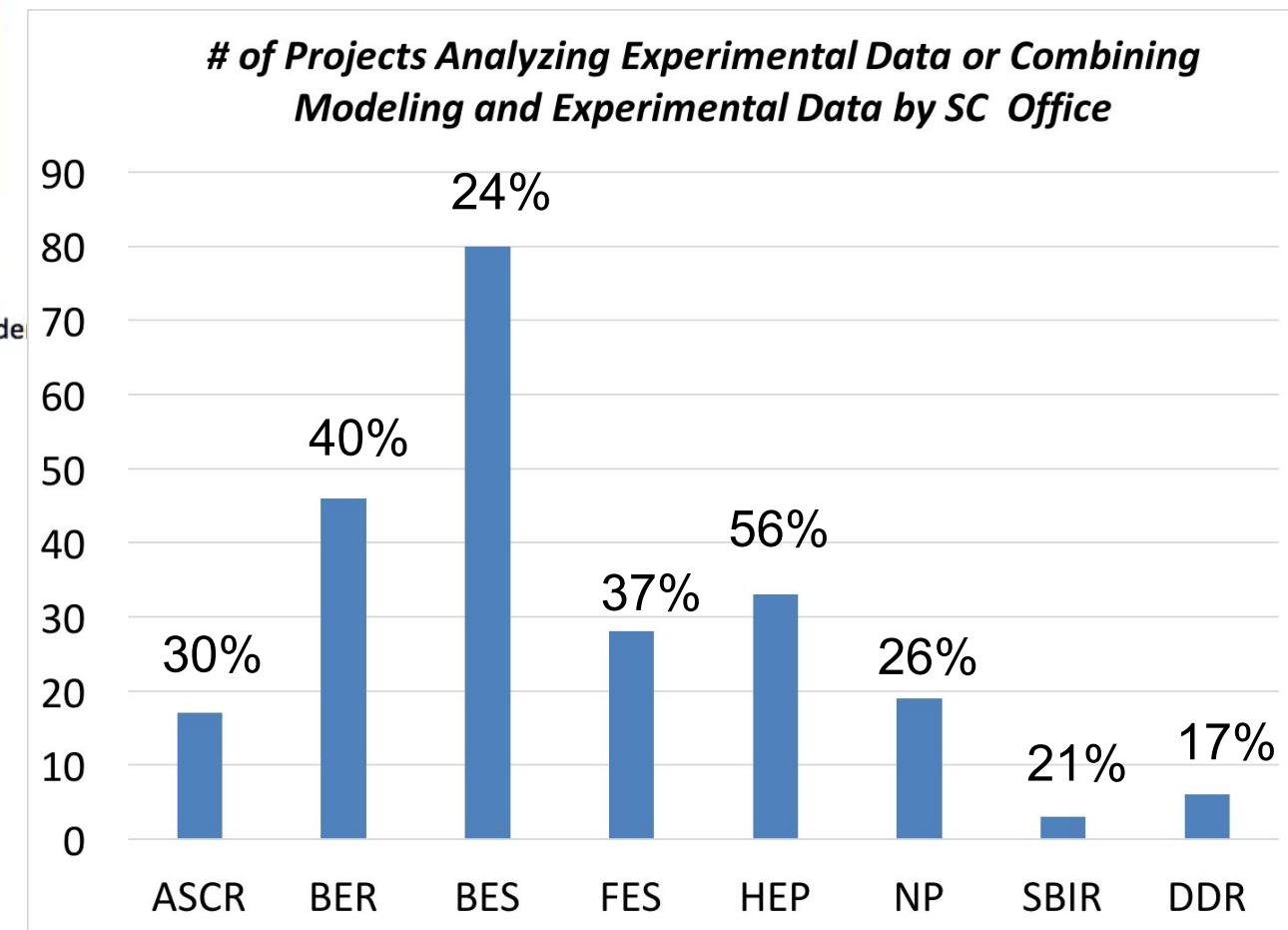
DESI



LSST-DESC

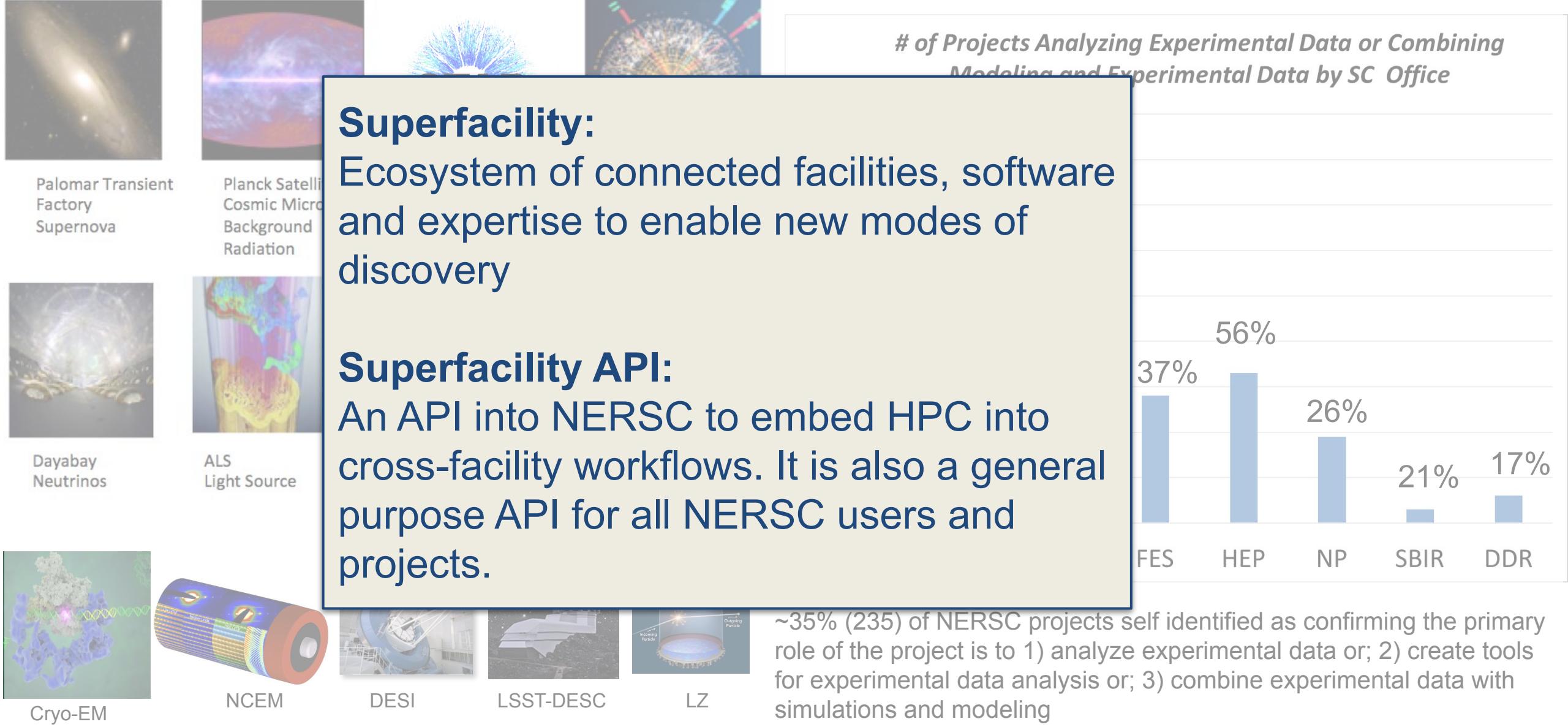


LZ



~35% (235) of NERSC projects self identified as confirming the primary role of the project is to 1) analyze experimental data or; 2) create tools for experimental data analysis or; 3) combine experimental data with simulations and modeling

NERSC supports a large number of users and projects from DOE SC's experimental and observational facilities



Why an API?

- **Meets a critical need; automation is no longer optional**
 - Unattended operation; minimizing HITL
 - Track/submit large number of jobs
 - Interface with collaborations, workflows and machines
- **NERSC becomes “machine readable”**
 - Enables easier creation of UIs, portals, etc.
 - Allows integration with control/analysis software
 - “ NERSC inside™ ”
- **Less DIY: simpler, standardized tooling (Python, etc)**
 - Stable refactor target for established projects or easier on-ramp for new ones
 - Contribute to HPC interface standards for portability
 - Authentication and security models

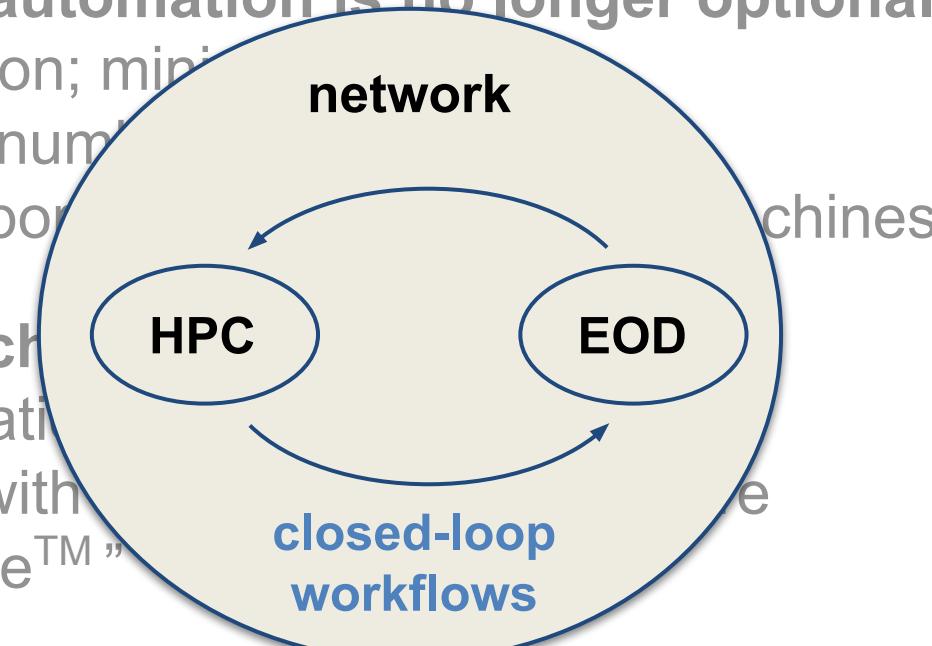
Drivers:

- Complex workflows
- Data-driven projects
- Real-time compute and streaming data from instruments
- Automation



Why an API?

- Meets a critical need; automation is no longer optional
 - Unattended operation; mini
 - Track/submit large numbers of jobs
 - Interface with collaborat
- NERSC becomes “machines”
 - Enables easier creation of complex workflows
 - Allows integration with external systems
 - “NERSC inside™”
- Less DIY: simpler, standardized tooling (Python, etc)
 - Stable refactor target for established projects or easier on-ramp for new ones
 - Contribute to HPC interface standards for portability
 - Authentication and security models



Drivers:

- Complex workflows
- Data-driven projects
- Real-time compute and streaming data from instruments
- Automation

What is the API good for?

Vision: all NERSC interactions are callable to facilitate seamless, automated "NERSC inside" workflows without a human in the loop.

Endpoints

- /status
- /account
- /compute
- /storage
- /tasks
- /utilities
-/beta

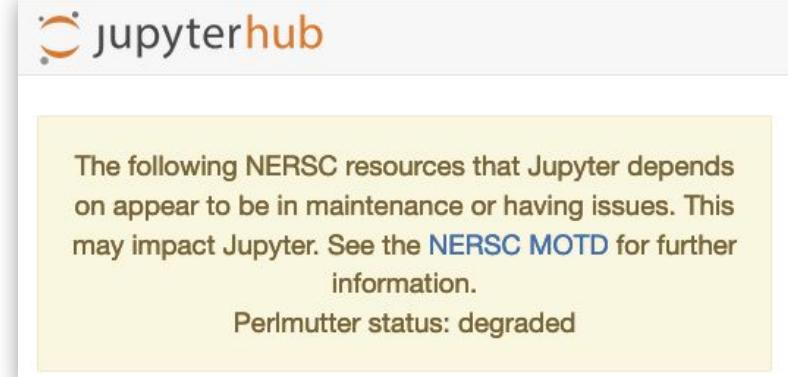
Example request

```
curl -X 'GET' \  
  'https://api.nersc.gov/api/v1.2/status/perlmutter' \  
  -H 'accept: application/json'
```

Result:

```
{  
  "name": "perlmutter",  
  "full_name": "Perlmutter",  
  "description": "System is active",  
  "system_type": "compute",  
  "notes": [],  
  "status": "active",  
  "updated_at": "2023-03-02T18:00:00-08:00"
```

API is used both internally at NERSC and externally for our users



Interface documentation

<https://api.nercsc.gov/api/v1.2>

- Interactive, up-to-date and self-documenting
- See endpoints, payloads, example code
- Works with any dev environment
- Endpoints with significant work are asynchronous
- End user docs and examples:
<https://docs.nercsc.gov/services/sfapi/>

Authorization

- Some endpoints are public and don't need an access token (no lock icon)
- For authorized endpoints (with lock icon), click the "Authorize" button at the top page and paste the access token



NERSC SuperFacility API 1.2 OAS3

/api/v1.2/openapi.json

A programmatic way to access resources at [NERSC](#)
For information on how to authenticate and use the api, please see the [documentation](#)

[Terms of service](#)
[NERSC Contacts - Website](#)

Servers /api/v1.2 [Authorize](#)

meta Information about this Superfacility API installation

status NERSC component system health

account Get accounting information about the user's projects

GET	/account/projects	Read Projects	▼	🔒
GET	/account	Read User	▼	🔒
GET	/account/roles	Read Roles	▼	🔒
GET	/account/groups	Read Groups	▼	🔒
POST	/account/groups	Create Group	▼	🔒
GET	/account/groups/{group}	Read Group	▼	🔒
PUT	/account/groups/{group}	Update Group Membership	▼	🔒

compute Run commands and manage batch jobs on NERSC compute resources

GET	/compute/jobs/{machine}	Read Jobs	▼	🔒
POST	/compute/jobs/{machine}	Submit Job	▼	🔒

Projects using the SF API

Since release in 2022 (Feb 2023 numbers)

- 27 non-staff users made clients
- members of 40 different non-staff projects.

~ 12M logged requests from May 2022 - Feb 2023
= one request every 2 sec

Examples



automated data processing for serial diffraction/scattering workflows



automated processing (HTC) of 4D STEM scan data with the [Distiller](#) app.



automated data processing between each of their plasma shots to reconstruct the shape and properties of the plasma.

Jupyter at NERSC

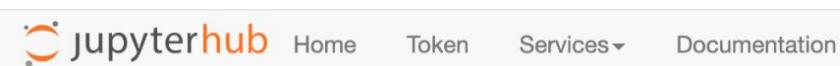
- Hosted at jupyter.nersc.gov
- Invaluable component of NERSC's data intensive science portfolio
 - Data cleaning and data transformation
 - Numerical simulation
 - Machine learning
 - Workflows and analytics frameworks
 -
- Configurable notebook servers
 - Shared, CPU/GPU, reservations
- Comes with NERSC's conda env
 - but run your own conda env,, container etc.



A screenshot of a Jupyter Notebook interface. The top navigation bar includes File, Edit, View, Run, Kernel, Tabs, Settings, and Help. The left sidebar shows a file tree with notebooks like Data.ipynb, Fasta.ipynb, Julia.ipynb, R.ipynb, iris.csv, lightning.json, and lorenz.py, along with a terminal tab, a console tab, and a code tab. The main area displays a Lorenz attractor plot with three sliders for sigma (10.0), beta (2.67), and rho (28.0). Below the plot is a code cell in Python 3:

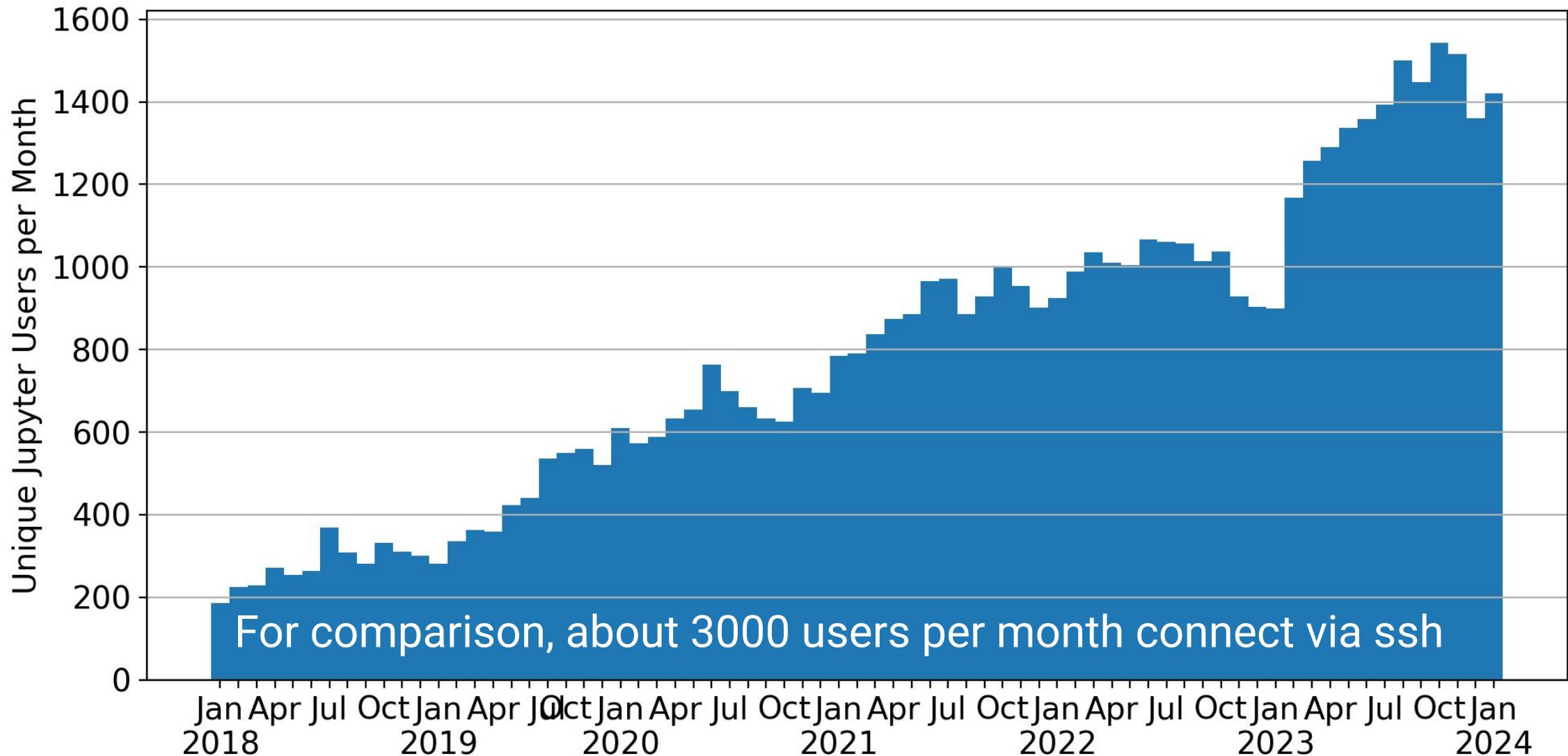
```
In [4]: from lorenz import solve_lorenz
t, x_t = solve_lorenz(N=10)
```

The output view shows the Lorenz attractor plot. The code cell below it contains the definition of the `solve_lorenz` function, which plots the Lorenz differential equations. The plot shows two trajectories swirling around two points.



	Shared CPU Node	Exclusive CPU Node	Exclusive GPU Node	Configurable Job
Perlmutter	start	start	start	start
Resources	Use a node shared with other users' notebooks but outside the batch queues.	Use your own node within a job allocation using defaults.		Use multiple compute nodes with specialized settings.
Use Cases	Visualization and analytics that are not memory intensive and can run on just a few cores.	Visualization, analytics, machine learning that is compute or memory intensive but can be done on a single node.		Multi-node analytics jobs, jobs in reservations, custom project charging, and more.

Jupyter Usage at NERSC



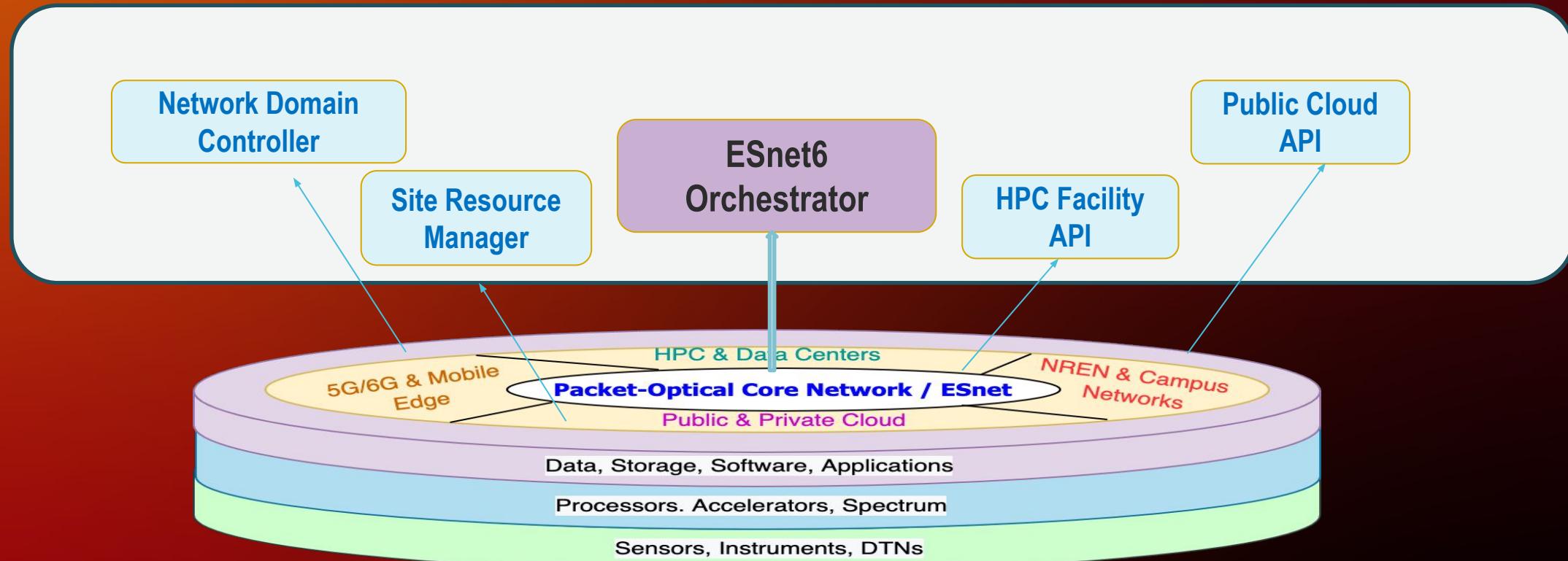
Lightning Talk II

Cross-Facility Science Workflows Automation and Orchestration

Xi Yang
Energy Sciences Network (ESnet)



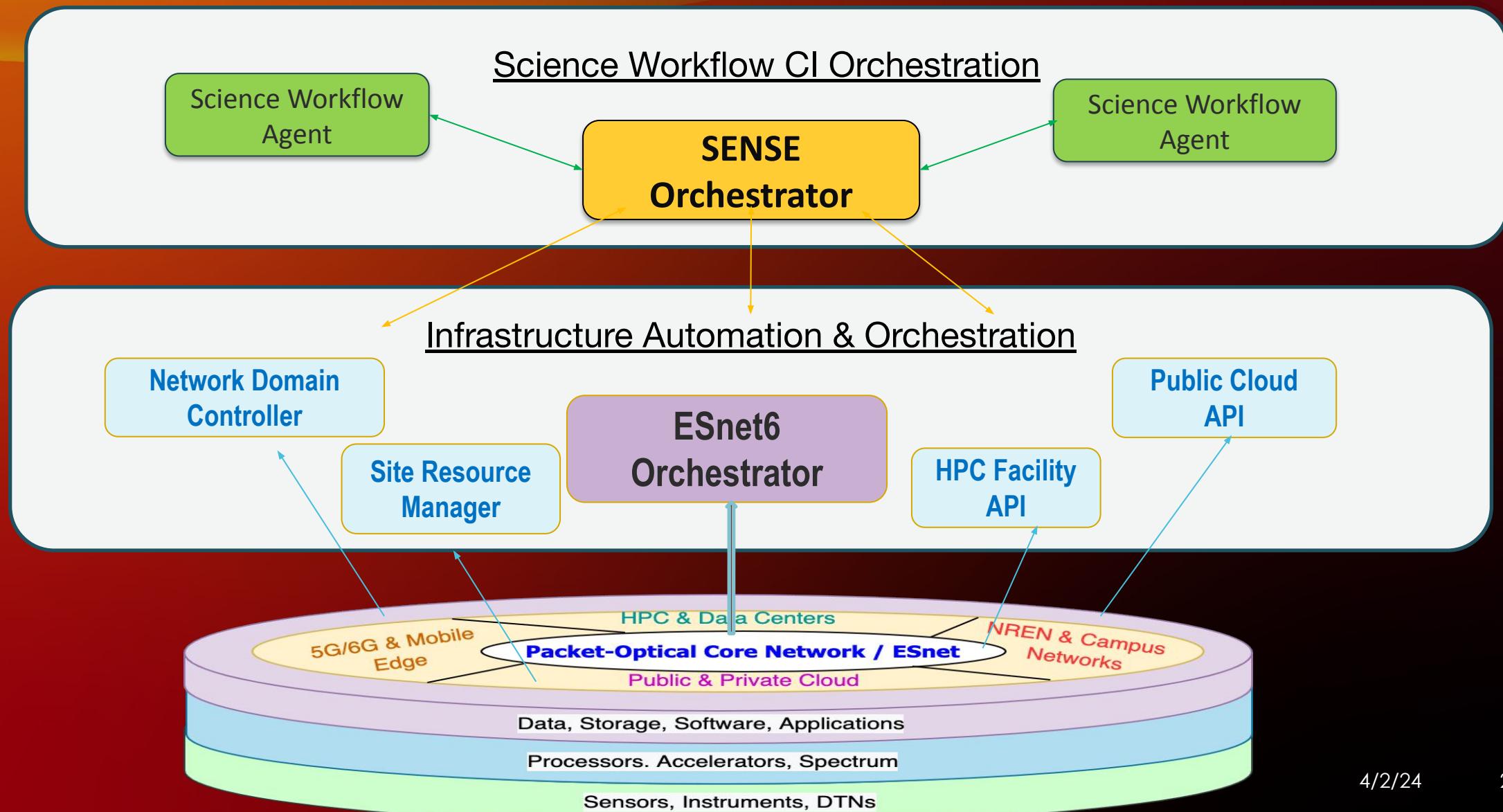
Facility Infrastructure Automation and Orchestration



Cross-Facility Fundamentals

- **Connection**
 - Foundation level: L1 / L2 / L3
- **Identity**
 - Federated IDs to access native and external resources.
- **Facility API**
 - Facilities may use common APIs and common information models
 - Enforcement and maintenance of API interworking can be challenging
- **Workflow Tool**
 - Besides common API, workflow tools may understand multiple APIs, allocate resources on multiple facilities, and create connections between them.

Cross-Facility Science Workflows Automation and Orchestration



SENSE - An Example API-Based Cross-Facility Architecture

<https://sense.es.net>

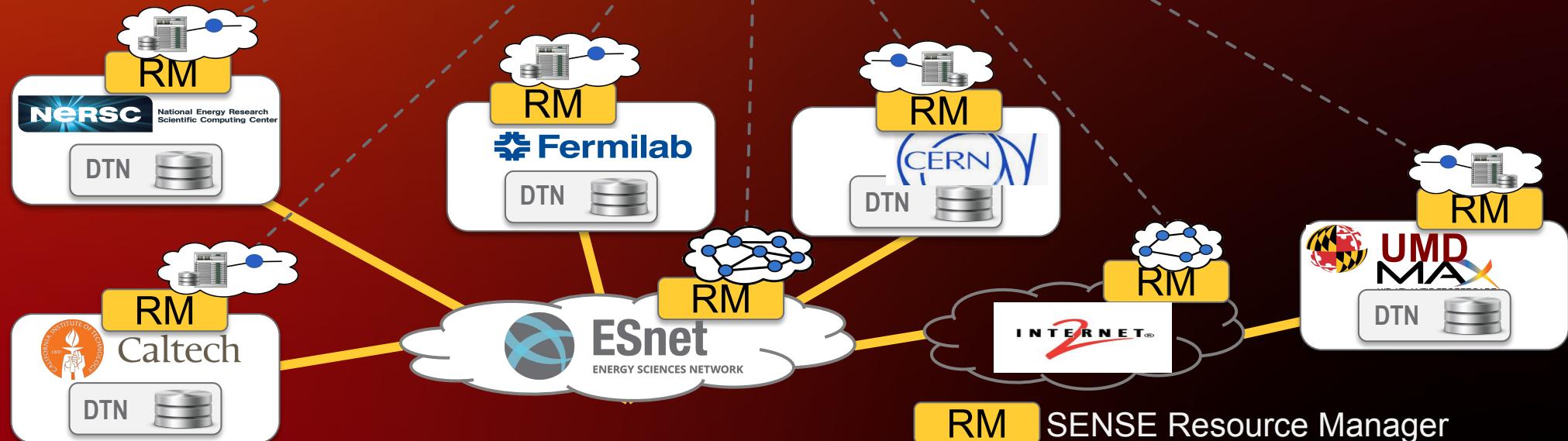
Application Workflow Agent

Intent-based APIs with resource discovery, negotiation, service lifecycle management & monitoring

Real-time system based on Resource Manager developed infrastructure and service models

SENSE Orchestrator

SENSE End-to-End Model



SENSE Recommended Best Practices for Cross-Facility API

- “Functional” API is more important than “same” API
- Deterministic, negotiable and reservable services are the key
- Provides consistent resource model, service and policy definitions
- Offload orchestration intelligence to middleware
- Middleware normalizes your interface with the “plural” APIs
- Workflow agents Interact with middleware by iterating on what-if questions, agreements and feedbacks on behalf of user applications

Extrapolation – The Questions to Ask

- **API Structure**

- What does a “Common API” means?
- Is it a single specification, a set or a hierarchy of APIs?
- Is north-south API interaction between workflow and facilities enough or we also need east-west facility interactions?

- **Information Model**

- Can we build an ecosystem of universal ontologies and semantics?
- Can we agree on a common resource modeling language and method?
- What are the most important common functions?

- **Access Policy**

- Can we normalize user identity and attributes across facility IdPs?
- Can we represent / translate workflow policy to facility native policies?
- Do we need a global clearinghouse? (enforcement)

Lightning Talk III

Services for integrating Data Portals

Ryan Prout

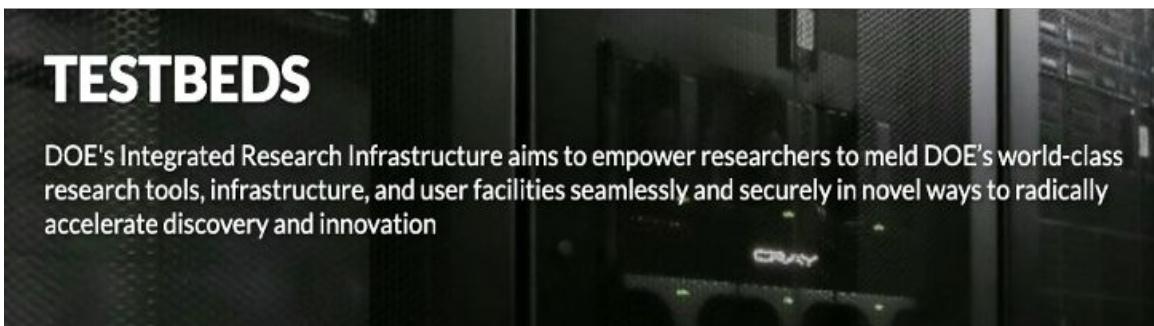
Oak Ridge Leadership Computing Facility (OLCF)



What is the OLCF?

The OLCF supports research in biology, chemistry, seismology, engineering, energy, and many other fields. **OLCF simulations have improved the safety and performance of nuclear power plants, turbomachinery, and aircraft; accelerated development of new drugs and advanced materials; and informed design of an international fusion reactor.** The simulations have explored hurricanes, biofuels, neurodegenerative diseases, and clean combustion for power and propulsion.

Multi-enclave HPC ecosystem



Defining different avenues of exploration (an HPC Facility perspective)

Facility-developed portals and workflow orchestration systems

- Objective is to enhance the local and unique facility ecosystem

Externally-developed data portals and workflow orchestration systems

- Objective is to extend the local facility ecosystem to an external system

Augment the HPC experience and large-scale simulation workflow, from running jobs to sharing data and insight with a larger community

Operational challenges of externally developed data portals and workflow orchestration systems (an HPC Facility perspective)

Policy and New Modes of Trust

Integration and access

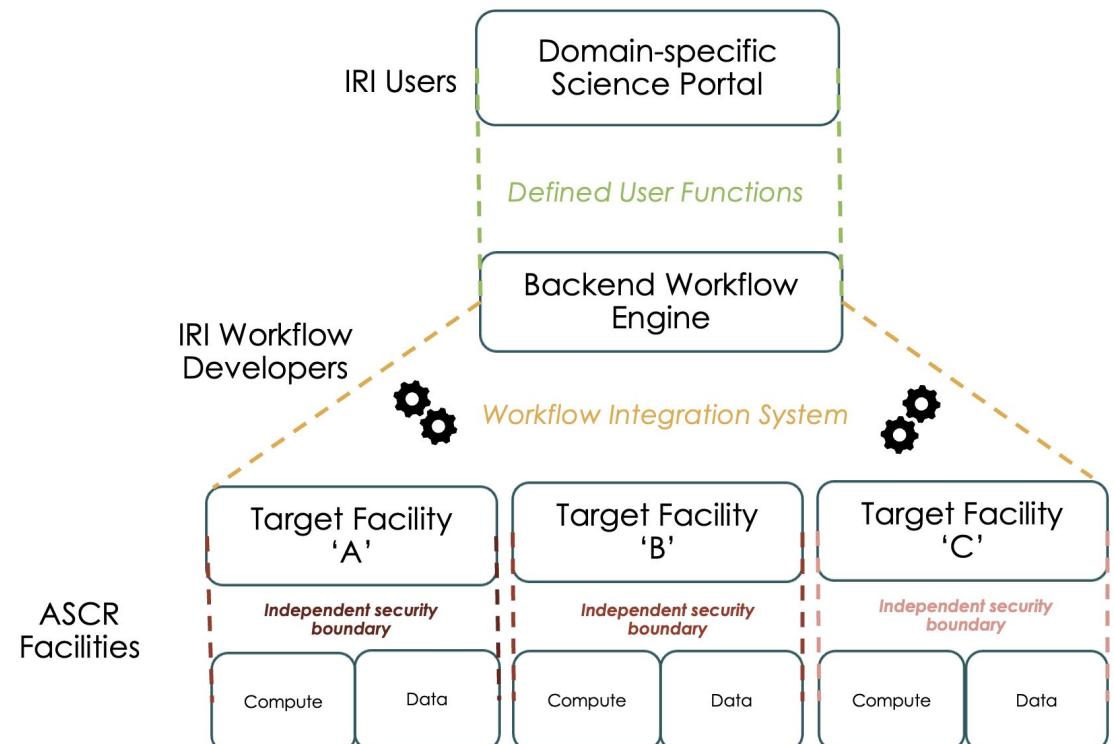
- Emphasis on software-to-software interaction (APIs and services instead of users and SSH sessions)
- Authentication and authorization
 - How do we trust external users and authorize them through third-party workflow systems?

Software Engineering

Extending access to protected facility resources

- Extend access to facility provided resources to the outside world in new and secure ways
- Meet policy requirements
- External developer experience and user experience

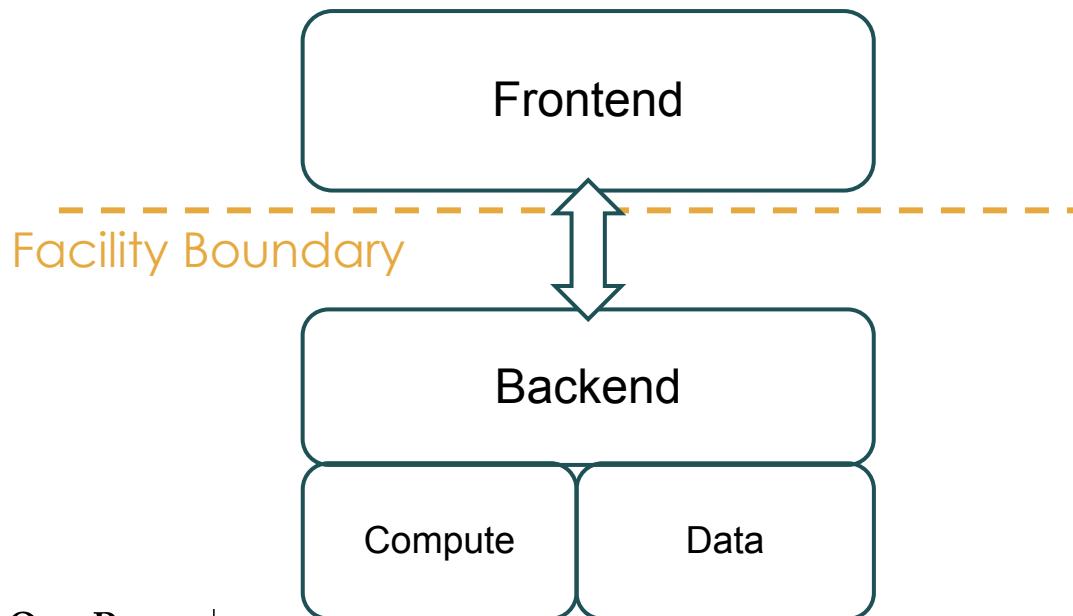
Challenges increase as you go across facilities. This is the IRI solution space.



Arising IRI patterns for data portals

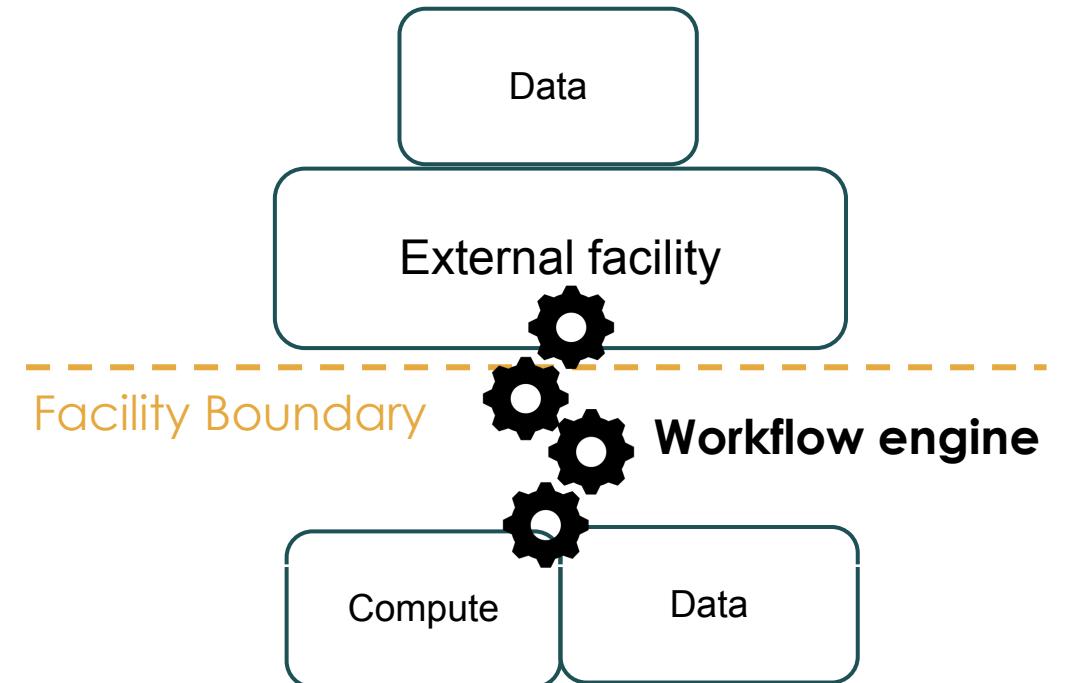
Server-side computing

- Backend provides data and limited compute functions
- Frontend users are limited to specific actions on provided data



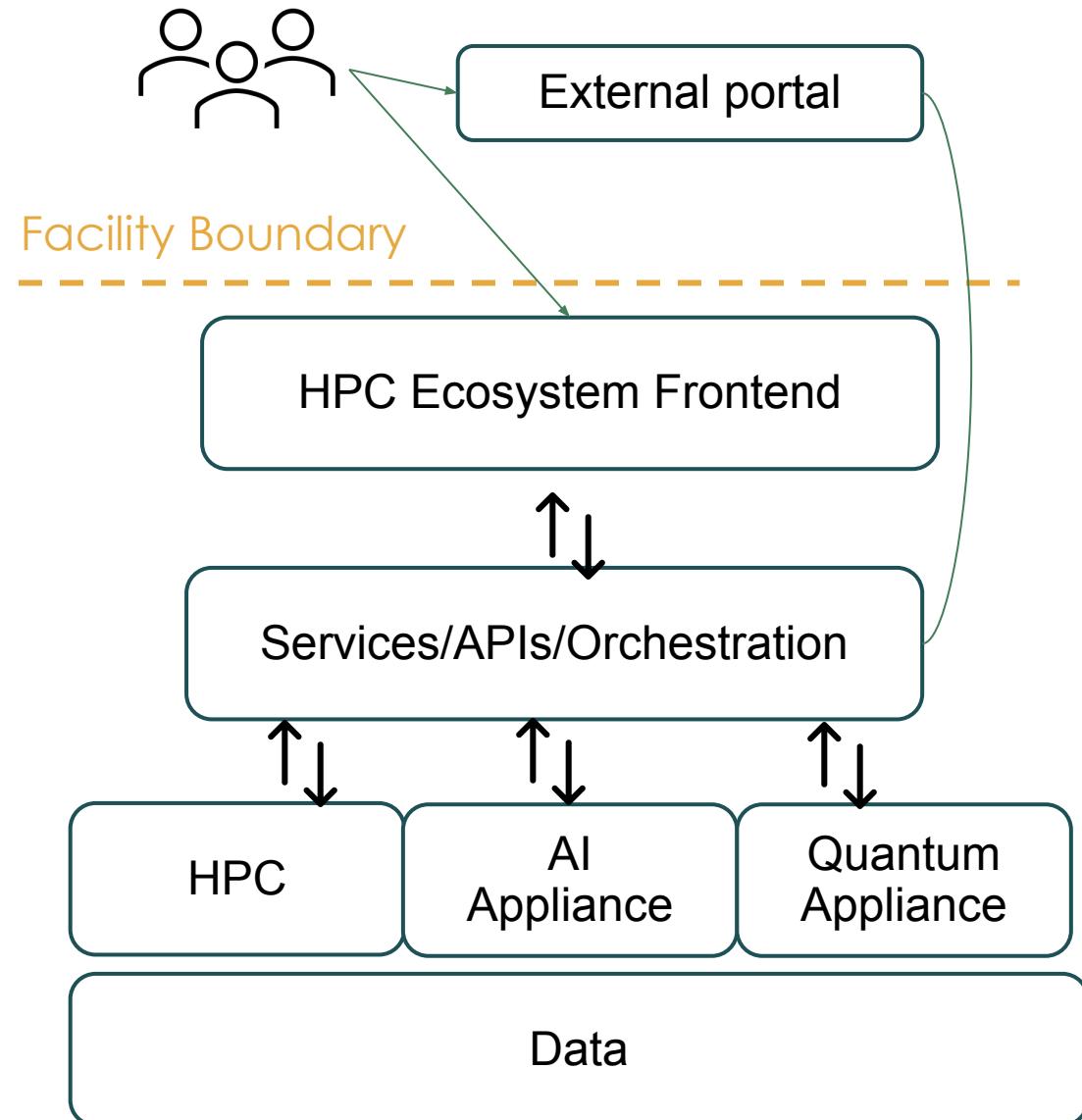
Facility-to-facility

- Data resides at external facility
- Users of external facility utilize a provided workflow engine to move data and access remote compute resources



Exploration of a software-driven facility ecosystem

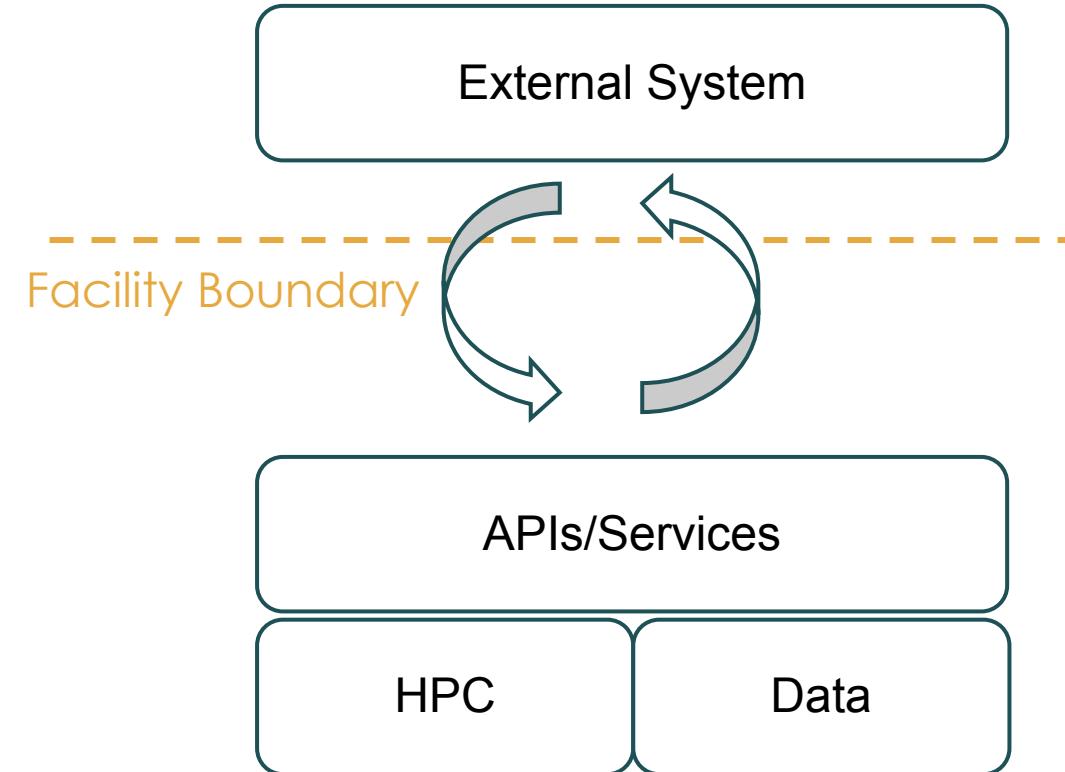
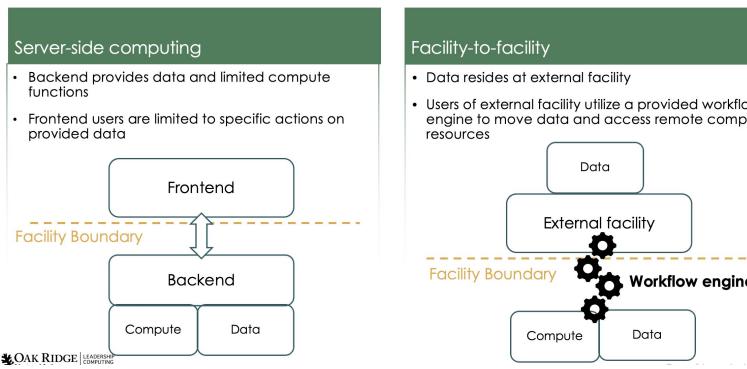
- Decouple systems from services and orchestration layer
- Services and orchestration layer provides access to an array of backend systems
- Decoupling allows services and orchestration to happen "above" target HPC/AI/Quantum systems
- HPC Frontend connects to controlled APIs that drive workflows on the backend systems
- Facility staff can develop and integrate services that plug into the middle layer in support of the diverse user base



Exploration of services to support external integration

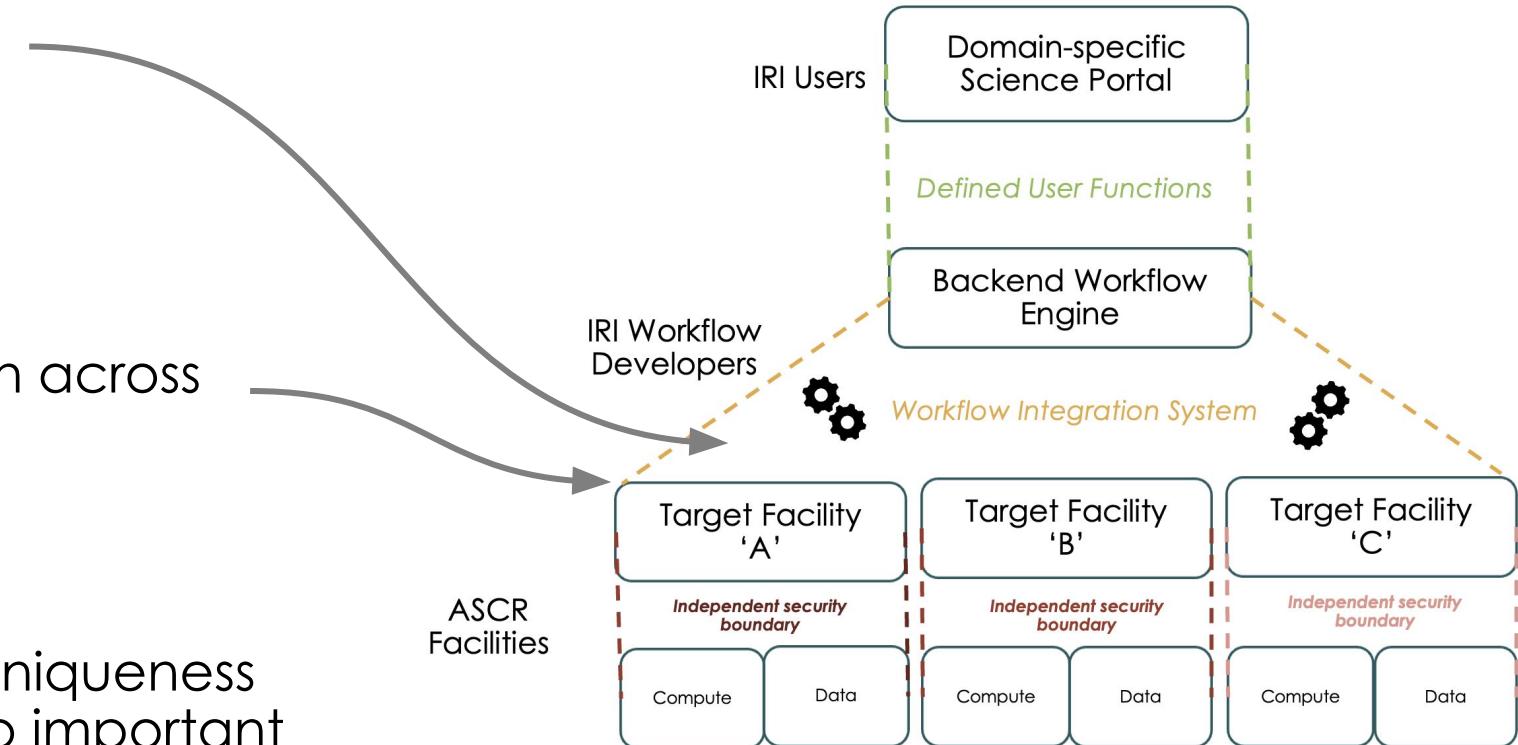
- Operational requirements are still being defined
 - Operational trust between the facility and the external system are needed
- We are exploring a Facility API system and Globus services
 - Interested in how these systems can complement each other
 - Important for ASCR Facilities to provide options
 - AuthN/AuthZ policies for APIs and tokens
- Token management is key to the user experience
 - Can we enable long-lived tokens to external portals with very understood and limited access to facility resources?
 - Globus tokens vs. Facility tokens?
- Identifying certain patterns arising is a key first step:

Arising IRI patterns for data portals



ResearchOps for IRI

- What do IRI developers need?
 - Policy evolution
 - Technical capabilities
- What should facilities prioritize?
- How do we streamline research across ASCR?
 - People
 - Tools and Infrastructure
- Important to maintain facility uniqueness in terms of capabilities, but also important to drive towards a common IRI user experience



Lightning Talk IV

Globus Platform Interfaces for Automating Science

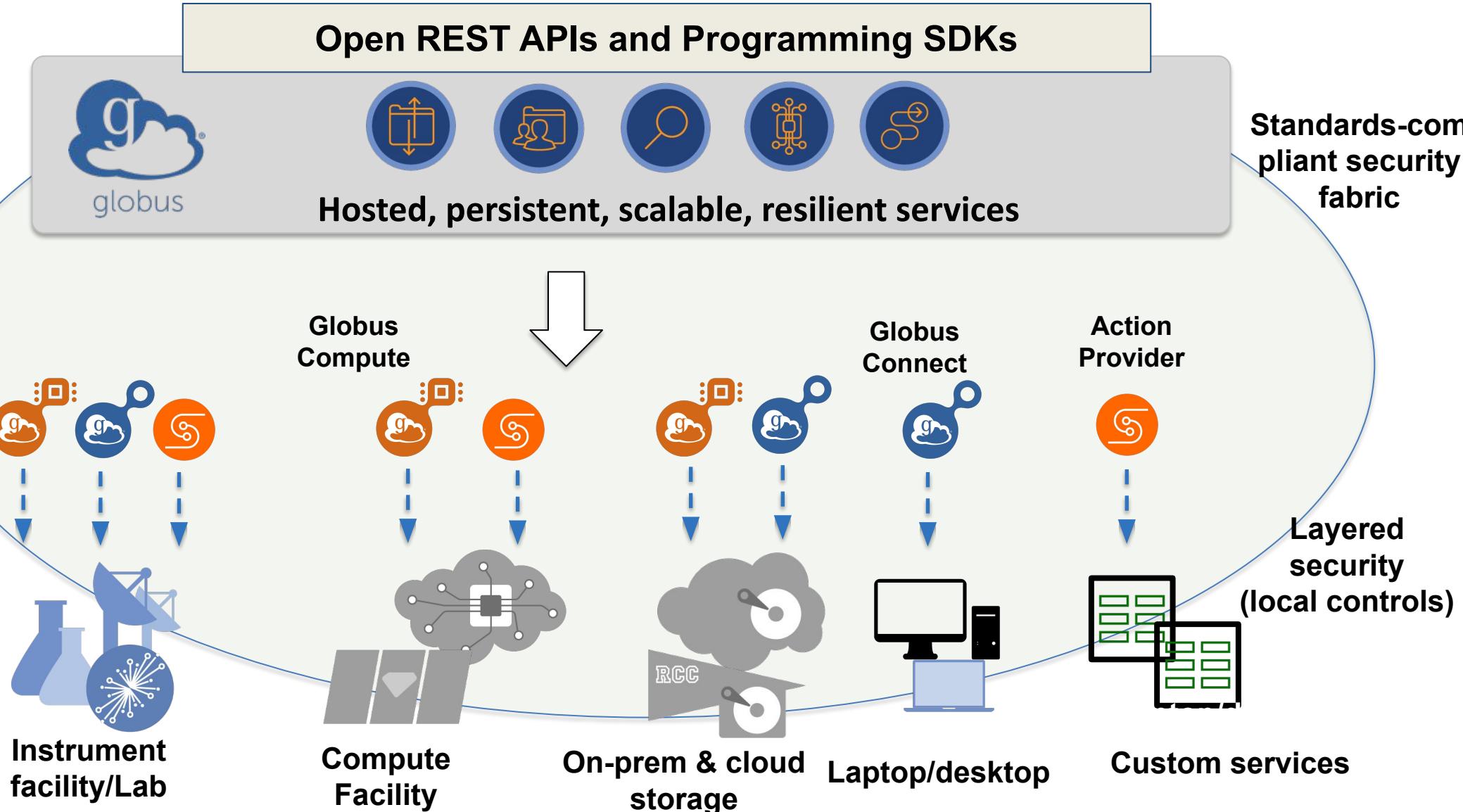
Kyle Chard
Argonne National Lab (ANL)





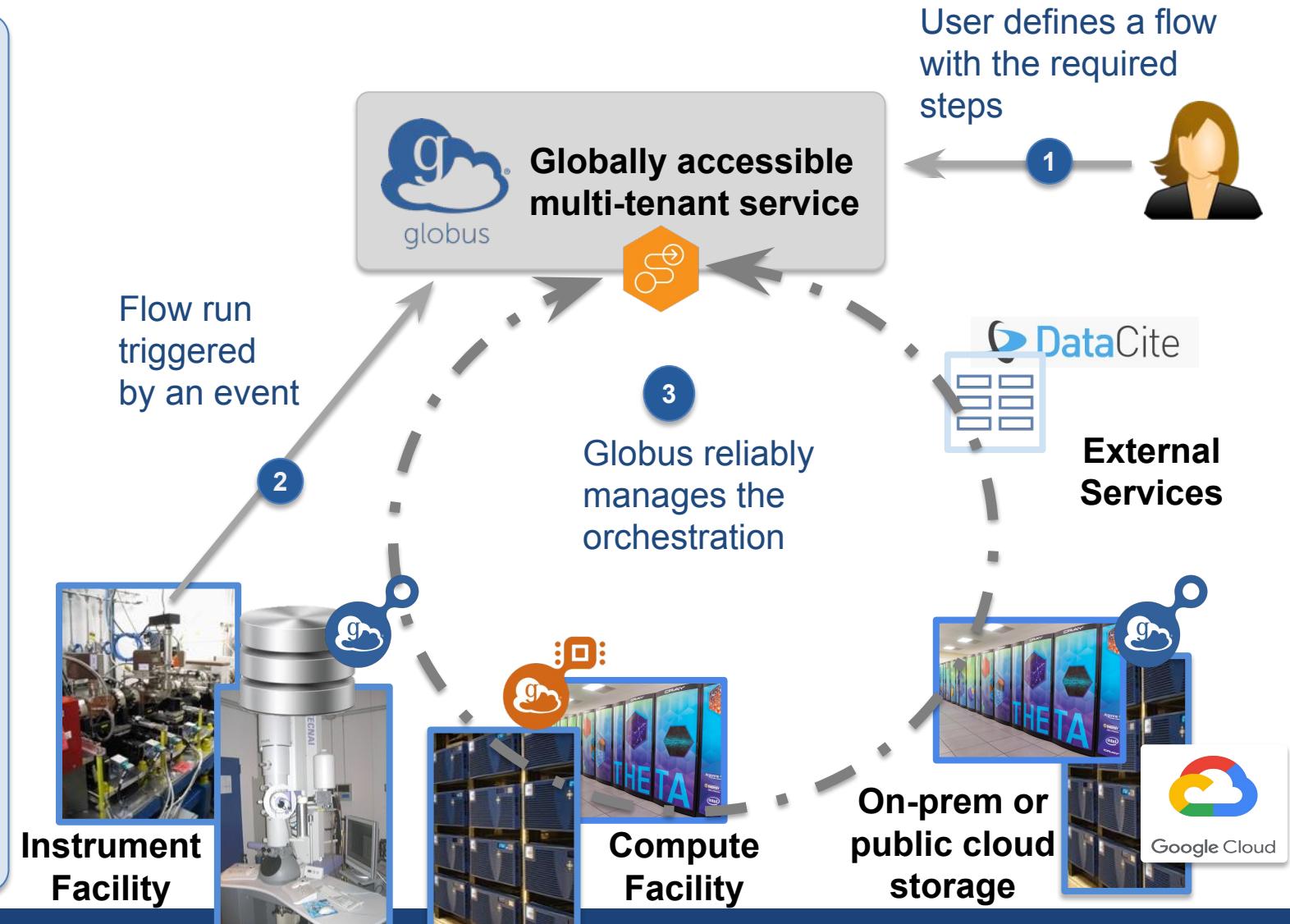
Globus implements a hybrid model for research IT

Global management & orchestration



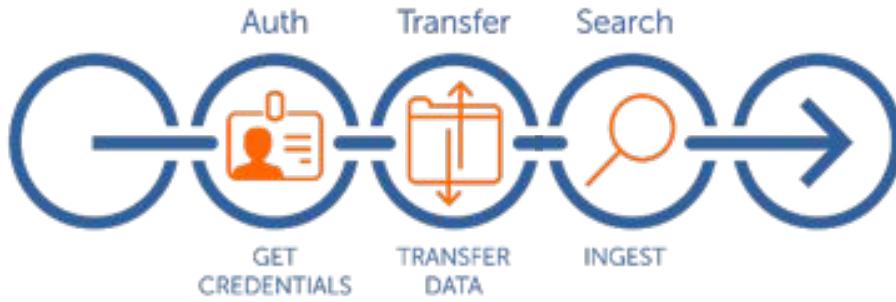
Automating research with Globus Flows

- Managed reliable task orchestration
- Event driven execution model – state machine
- Declarative language for flow definition
- Extensible for integrating external services via a common asynchronous action API
 - /run
 - /status
 - /cancel
 - /release

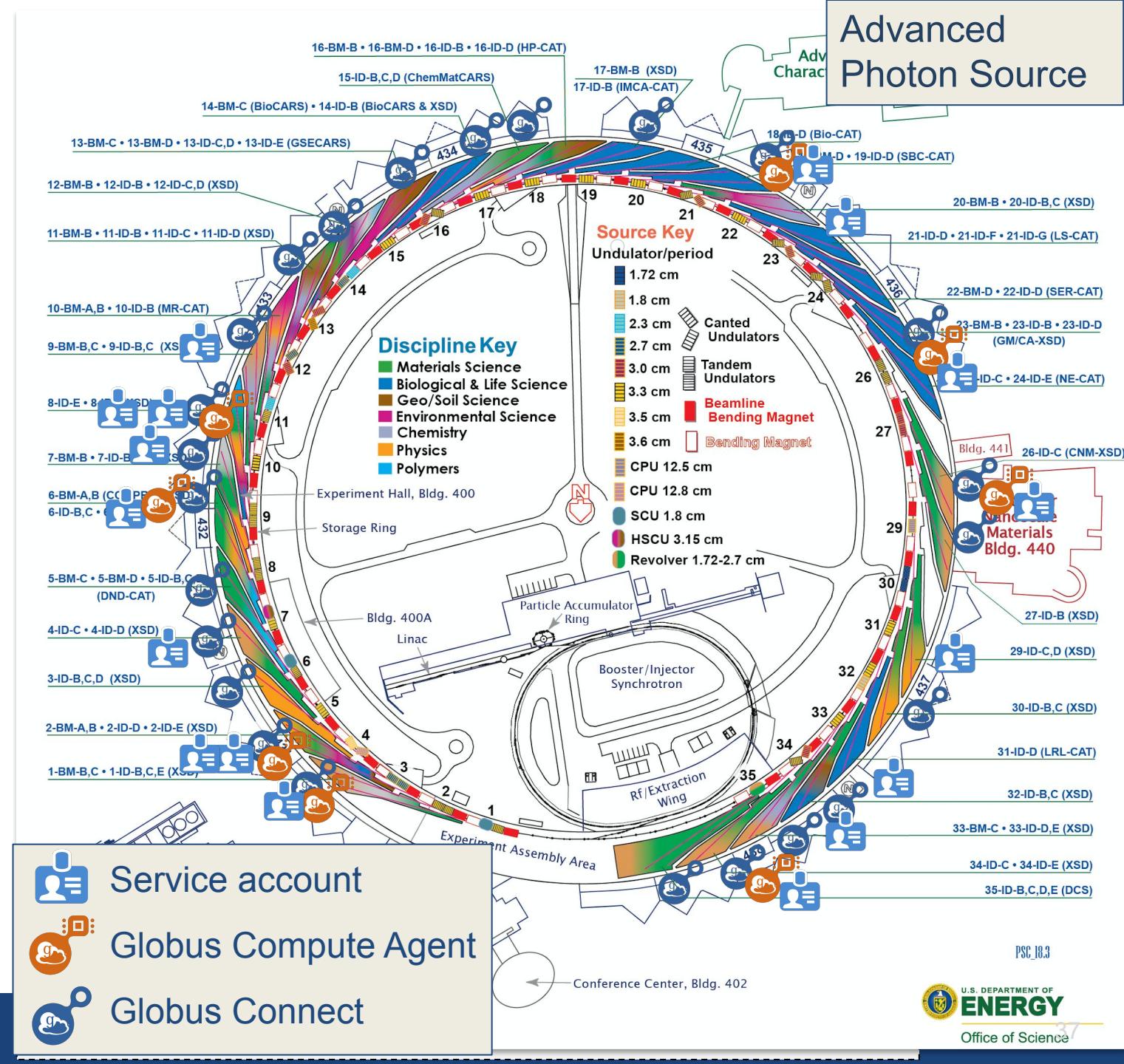
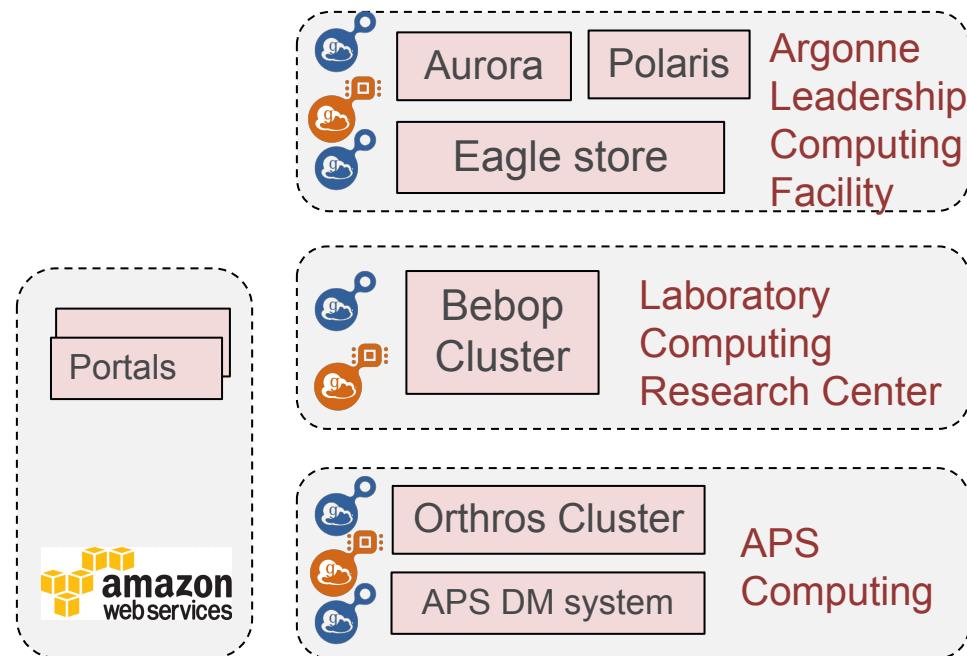




APS and ALCF

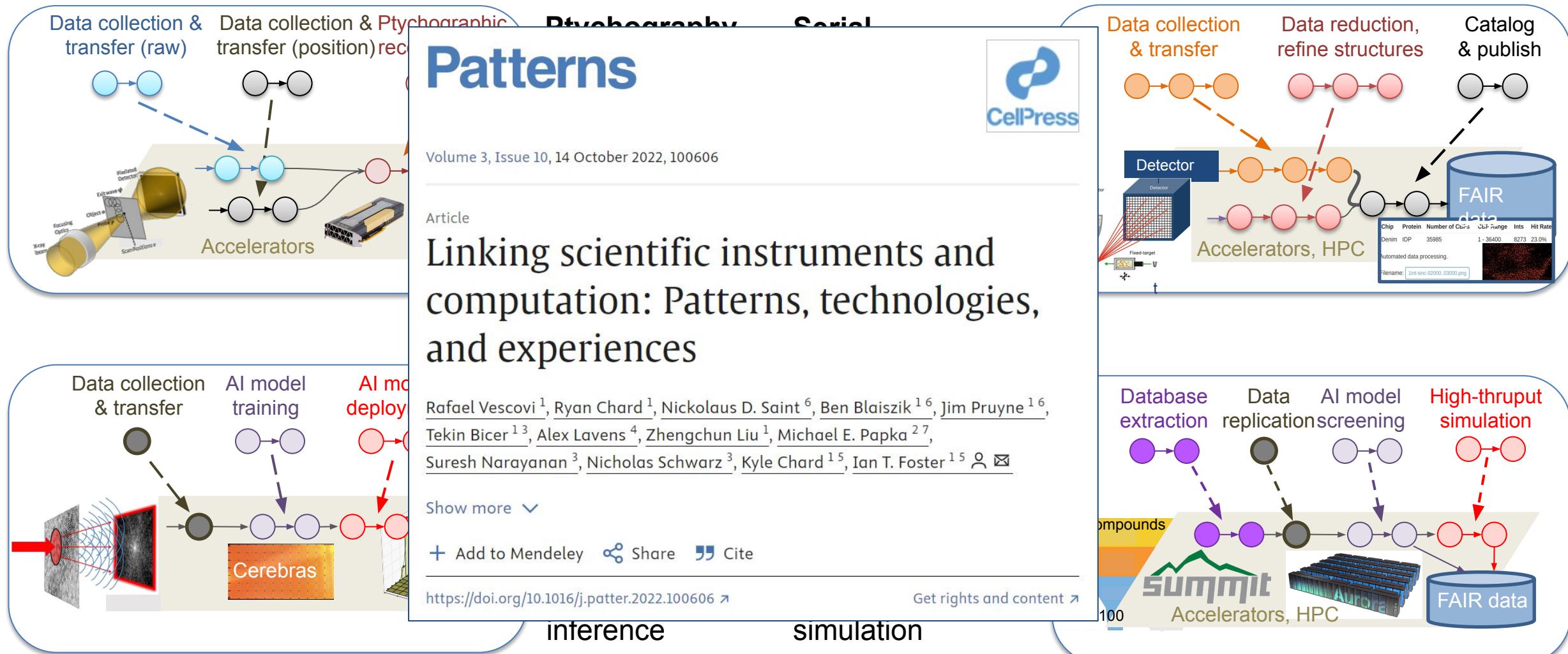


~20 service accounts with
On-demand access to 56 Polaris nodes





Linking scientific instruments and computation

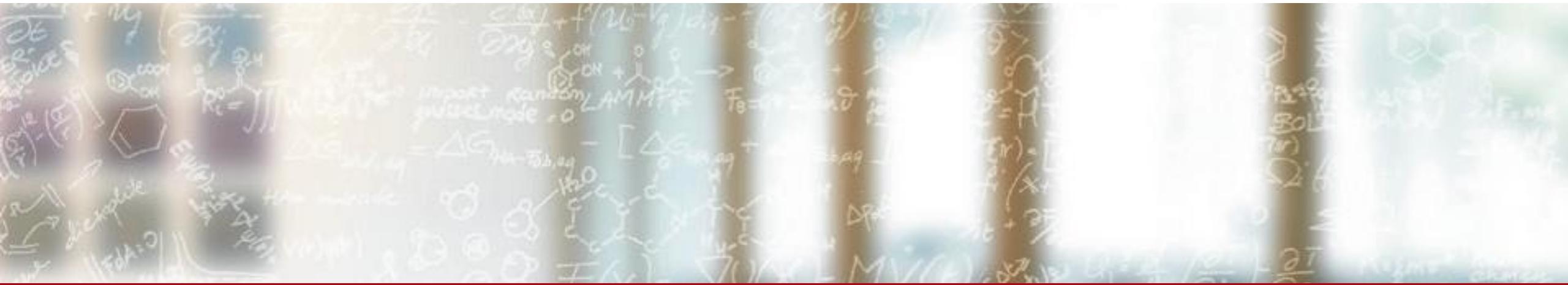


Lightning Talk V

FirecREST: an Open-Source API for HPC

Juan Pablo Dorsch
CSCS





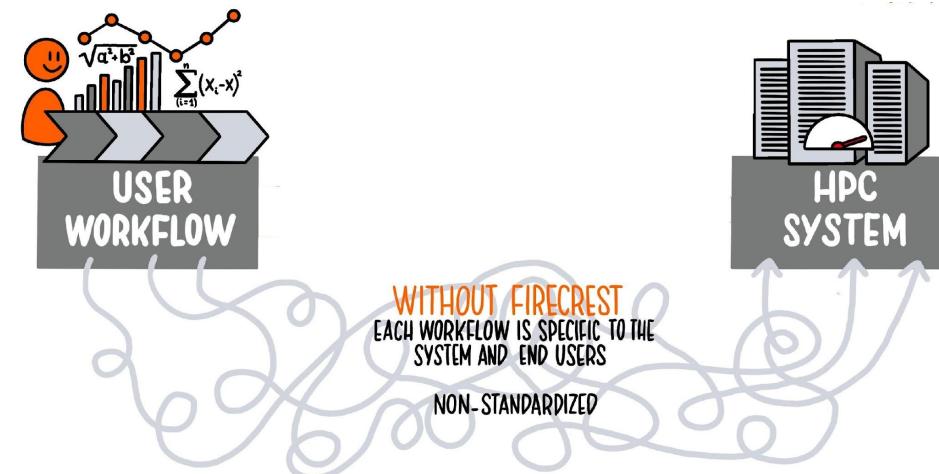
FirecREST: an open-source API for HPC

Juan Pablo Dorsch (juanpablo.dorsch@cscs.ch)
CSCS - Swiss National Supercomputing Centre
ETH-Zürich

November 21, 2024
SC24
Atlanta, GA, USA

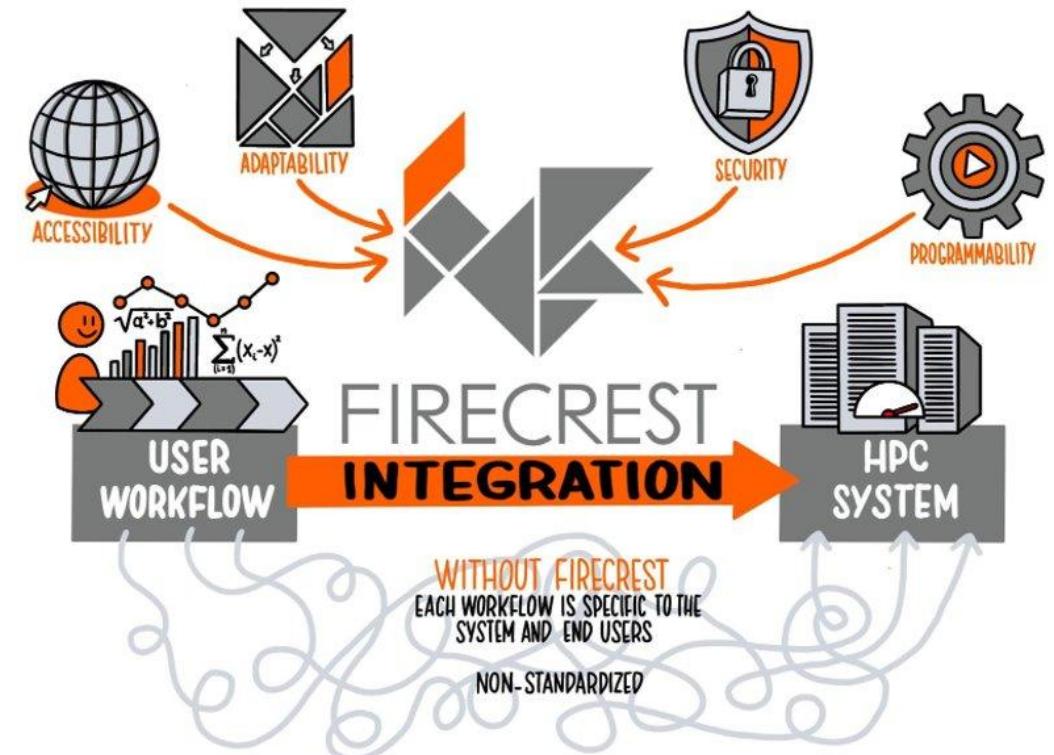
Motivation

- **User community needs:** the HPC user community increasingly requires automated workflows and access to advanced services to streamline their computational tasks
- **Infrastructure Challenges:** Supporting this diverse range of services and workflows poses significant scalability challenges for HPC infrastructure
- **Programming Standards:** These demands highlight the need for standardized programming practices to ensure efficient and consistent usage of HPC resources.



FirecREST features

- FirecREST is an open-source web-enabled API to HPC resources developed by CSCS
- Includes abstractions for workload managers and schedulers, data transfer, and common filesystem's operations
- Allows workflow execution and data transfer across supercomputing facilities
- Facilitates workflows support, monitoring and observability



The FirecREST API

- OpenAPI documentation: <https://firecrest-api.csccs.ch>

The screenshot displays the OpenAPI documentation for the FirecREST API, organized into three main sections: Status, Compute, and Storage.

Status (Left Column): Status information of infrastructure and services.

- GET /status/services** List of services
- GET /status/services/{servicename}** Get service information
- GET /status/systems** List of systems
- GET /status/systems/{machinename}** Get system information

Utilities (Bottom of Status): Basic system utilities. All calls are blocking and low-latency operations, mostly used for administrative tasks.

- GET /utilities/ls** List directory contents
- POST /utilities/mkdir** Creates a directory
- PUT /utilities/rename** Rename/move a file, directory, or symlink
- PUT /utilities/chmod** Change file mode bits
- PUT /utilities/chown** Change file owner and group
- POST /utilities/copy** Copy file from a filesystem path to another
- GET /utilities/file** determine file type
- GET /utilities/head** Prints first part of a file
- GET /utilities/stat** determines the status of a file

Compute (Top Right): Non-blocking calls to workload manager to submit and query jobs. The service responds with a JSON object containing job information.

- POST /compute/jobs/upload** Submit Job by uploading a local sbatch file
- POST /compute/jobs/path** Submit Job by a given remote sbatch file
- GET /compute/jobs** Retrieves information from all jobs
- GET /compute/jobs/{jobid}** Retrieves information about a specific job
- DELETE /compute/jobs/{jobid}** Delete Job
- GET /compute/acct** Job account information

Storage (Bottom Right): Non-blocking calls to high-performance storage services. The service responds with a JSON object containing storage information.

- POST /storage/xfer-internal/rsync** rsync
- POST /storage/xfer-internal/mv** move (rename) files
- POST /storage/xfer-internal/cp** copy files and directories
- POST /storage/xfer-internal/rm** remove files or directories
- POST /storage/xfer-external/upload** Upload a file
- POST /storage/xfer-external/download** Download a file
- POST /storage/xfer-external/invalidate** Invalidate temporary URL

pyFirecREST Library

- pyFirecREST is a Python library that wraps FirecREST endpoints for scripting
- Seamless integration with OIDC/OAuth2 for JWT Access Token
- **Facilitates integration with tools that expose APIs or SDKs via Python or scripting languages**

pyfirecrest 2.7.0

pip install pyfirecrest

pyFirecrest is a python wrapper for FirecREST

Navigation

- Project description
- Release history
- Download files

Verified details ✓
These details have been *verified by PyPI*

Maintainers

eirinik

Project description

PyFirecREST

This is a simple python wrapper for the [FirecREST API](#).

How to install

- Through [PyPI](#):

```
python3 -m pip install pyfirecrest
```



Link to pyFirecREST Docs



CSCS

Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETHzürich

FirecREST to streamline services on HPC

FirecREST to streamline services on HPC

- Continuous Integration (CI) Pipelines
 - Running pipelines from the cloud using any CI/CD service
 - Avoids the need for SSH connection to the facility

A screenshot of a GitHub Actions run summary page. The page shows a successful workflow named "Rename secret vars #11". On the left, there's a sidebar with links for "Summary", "Jobs", "Run details", "Usage", and "Workflow file". The main area displays a table titled "Run and billable time" with the following data:

Job	Run time	Billable time
test_CSCS (daint)	10s	0s
	10s	0s

FirecREST to streamline services on HPC

- Interactive Computing
 - Jupyter Hub's FirecREST Spawner reduces the requirement on the HPC infrastructure side in terms of administration, machine provisioning, networking, etc.
 - Extending the well-known batchspawner allows to create JupyterLab Notebooks from the cloud.
 - The “recipe” can be replicated for several HPC systems by changing the configuration to a different system



Link to Jupyter's FirecREST
Spawner Docs

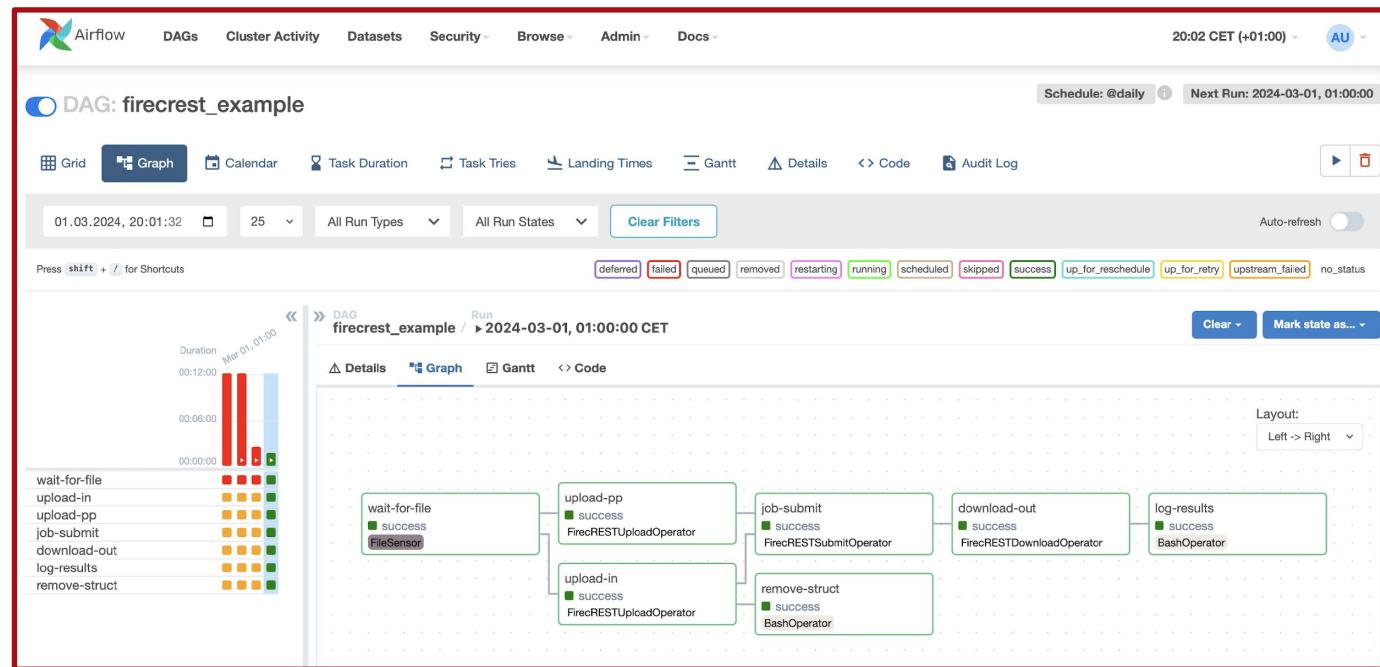
FirecREST to streamline services on HPC

- Scientific Portals
 - Successful use cases on integrating FirecREST and OpenOnDemand
 - Jupyter Notebooks and Filesystem plugins has been created using FirecREST
 - This way we can decouple OOD from the login node and the infrastructure of the HPC Center



FirecREST to streamline services on HPC

- Workflow Orchestrators
 - Enables an Airflow FirecREST Operator for HPC tasks (submit a job, upload/download data)
 - Create DAGs with those operators and run them from your laptop or the cloud



nextflow



FirecREST to streamline services on HPC

- User Portals

The screenshot shows the FirecREST Web UI v2 interface. On the left, there is a sidebar with the following navigation options:

- Dashboard
- Job Scheduler
- File Manager** (highlighted)
- Documentation
- Github repo
- Get support

The main area is titled "Filesystem" and shows the path: / capstor / scratch / cscs / jdorsch. A "Create directory" button is located in the top right corner. Below the path, there is a dashed box for file upload with the placeholder "Select a file or drag and drop". A "Upload file" button is located below this box. A table lists the contents of the directory:

Name	Last Modified	Size	Owner	Permissions	Actions
emptyfolder	04-11-2024 15:49:17	4 KB	csstaff	jdorsch	rwxr-x---
f7t_periodic_tests	18-11-2024 20:18:30	4 KB	csstaff	jdorsch	rwxr-x---
f7t_tests	06-06-2024 09:51:51	4 KB	csstaff	jdorsch	rwxr-x---
fc_jhub_tf-2.14.0.toml	07-03-2024 10:55:06	250 Bytes	csstaff	jdorsch	rw-r-----
fc_jhub.toml	16-09-2024 10:42:04	263 Bytes	csstaff	jdorsch	rw-r-----

At the bottom of the page, there is a copyright notice: "© 2024 CSCS - All rights reserved" and a version number: "Version 0.0.35".

FirecREST to streamline services on HPC

- User Portals

The screenshot shows the FirecREST Web UI v2 interface, specifically the Job Scheduler module. The left sidebar includes links for Dashboard, Job Scheduler (which is selected), File Manager, Documentation, Github repo, and Get support. The main area is titled "Job Scheduler" and displays a "List of jobs". There are four entries in the list:

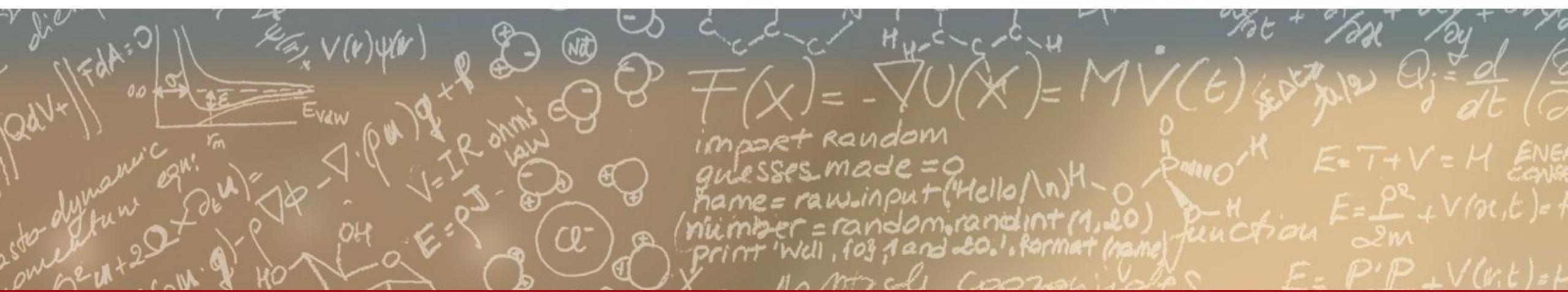
Job ID / Name	Status	Started On	Created By	Account	Nodes	Partition	Execution Time	Actions
Job ID: 1418212 / Name: f7t_test	CANCELLED by 24384	31-12-1969 19:00:00	jdorsch	csstaff	None assigned	debug	00:00:00	⋮
Job ID: 1418213 / Name: rm-job	COMPLETED	18-11-2024 04:25:53	jdorsch	csstaff	nid006795	normal	00:00:07	⋮
Job ID: 1419517 / Name: f7t_test	CANCELLED by 24384	31-12-1969 19:00:00	jdorsch	csstaff	None assigned	debug	00:00:00	⋮
Job ID: 1419518 / Name: rm-job	COMPLETED	18-11-2024 04:26:23	jdorsch	csstaff	nid006795	normal	00:00:07	⋮

At the bottom of the page, there is a footer with the text "© 2024 CSCS - All rights reserved" and "Version 0.0.35".

Links and references

- More on FirecREST
 - API Reference: firecrest-api.cscs.ch
 - FirecREST product page at CSCS: products.cscs.ch/firecrest
 - FirecREST public repository: github.com/eth-cscs/firecrest
 - FirecREST Docs (use cases): firecrest.readthedocs.io
 - pyFirecREST and CLI Docs: pyfirecrest.readthedocs.io
 - Join our community on Slack: firecrest-community.slack.com
 - Contact us: firecrest@cscs.ch





Thank you for your attention.

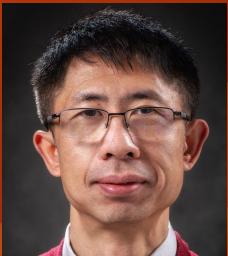
Q&A panel and open discussion



Bjoern Enders
NERSC/LBL



John MacAuley
ESnet



Xi Yang
ESnet



Ryan Prout
OLCF



Juan Pablo Dorsch
CSCS



Kyle Chard
ANL

